<<领域驱动设计>>

图书基本信息

书名：<<领域驱动设计>>

13位ISBN编号：9787115224071

10位ISBN编号：7115224072

出版时间：201004

出版时间：人民邮电出版社

作者：Eric Evans

页数：529

<<领域驱动设计>>

内容概要

领域模型使开发人员可以表达丰富的软件功能需求，由此实现的软件可以满足用户真正的需要，因此被公认为是软件设计的关键所在，其重要性显而易见。
但讲述如何将领域模型用于软件开发过程的优秀实用资料却不多见。
本书正是这一领域最著名的作品，受到众多业界大师的赞美和推介，广受读者好评。
　　要通过创建领域模型来加速复杂的软件开发，就需要利用大量最佳实践和标准模式在开发团队中形成统一的交流语言；不仅重构代码，而且要重构代码底层的模型；同时采取反复迭代的敏捷开发方法，深入理解领域特点，促进领域专家与程序员的良好沟通。
针对这些内容，本书结合真实项目，系统地介绍了领域驱动开发的目标、意义和方法，充分讨论了复杂系统的建模与设计问题。
　　本书将指导面向对象开发人员、系统分析人员和设计人员合理地组织工作，各有侧重、彼此协作，有条不紊地进行复杂系统的开发，帮助他们建立丰富而实用的领域模型，并由此创建长期适用的优质软件。

作者简介

Eric Evans世界著名软件建模专家，创建了Domain Language公司，致力于帮助公司机构创建与业务紧密相关的软件。
他在全球各地宣讲领域驱动设计的思想，开设课程、参加会议、接受专访，拥有大批的追随者。
从20世纪80年代开始，他就以设计师和程序员的双重身份参与过许多大型面向对象系统的设计和开发，涉及各种复杂的业务和技术领域。
同时，他还培训和指导过许多开发团队开展极限编程实践。

<<领域驱动设计>>

书籍目录

Part I Putting the Domain Model to Work 1 Chapter 1: Crunching Knowledge 7 Ingredients of Effective Modeling 12 Knowledge Crunching 13 Continuous Learning 15 Knowledge-Rich Design 17 Deep Models 20 Chapter 2: Communication and the Use of Language 23 UBIQUITOUS LANGUAGE 24 Modeling Out Loud 30 One Team, One Language 32 Documents and Diagrams 35 Written Design Documents 37 Executable Bedrock 40 Explanatory Models 41 Chapter 3: Binding Model and Implementation 45 MODEL-DRIVEN DESIGN 47 Modeling Paradigms and Tool Support 50 Letting the Bones Show: Why Models Matter to Users 57 HANDS-ON MODELERS 60 Part II The Building Blocks of a Model-Driven Design 63 Chapter 4: Isolating the Domain 67 LAYERED ARCHITECTURE 68 Relating the Layers 72 Architectural Frameworks 74 The Domain Layer Is Where the Model Lives 75 THE SMART UI “ANTI-PATTERN” 76 Other Kinds of Isolation 79 Chapter 5: A Model Expressed in Software 81 Associations 82 ENTITIES (A.K.A. REFERENCE OBJECTS) 89 Modeling ENTITIES 93 Designing the Identity Operation 94 VALUE OBJECTS 97 Designing VALUE OBJECTS 99 Designing Associations That Involve VALUE OBJECTS 102 SERVICES 104 SERVICES and the Isolated Domain Layer 106 Granularity 108 Access to SERVICES 108 MODULES (A.K.A. PACKAGES) 109 Agile MODULES 111 The Pitfalls of Infrastructure-Driven Packaging 112 Modeling Paradigms 116 Why the Object Paradigm Predominates 116 Nonobjects in an Object World 119 Sticking with MODEL-DRIVEN DESIGN When Mixing Paradigms 120 Chapter 6: The Life Cycle of a Domain Object 123 AGGREGATES 125 FACTORIES 136 Choosing FACTORIES and Their Sites 139 When a Constructor Is All You Need 141 Designing the Interface 143 Where Does Invariant Logic Go? 144 ENTITY FACTORIES Versus VALUE OBJECT FACTORIES 144 Reconstituting Stored Objects 145 REPOSITORIES 147 Querying a REPOSITORY 152 Client Code Ignores REPOSITORY Implementation; Developers Do Not 154 Implementing a REPOSITORY 155 Working Within Your Frameworks 156 The Relationship with FACTORIES 157 Designing Objects for Relational Databases 159 Chapter 7: Using the Language: An Extended Example 163 Introducing the Cargo Shipping System 163 Isolating the Domain: Introducing the Applications 166 Distinguishing ENTITIES and VALUE OBJECTS 167 Role and Other Attributes 168 Designing Associations in the Shipping Domain 169 AGGREGATE Boundaries 170 Selecting REPOSITORIES 172 Walking Through Scenarios 173 Sample Application Feature: Changing the Destination of a Cargo 173 Sample Application Feature: Repeat Business 173 Object Creation 174 FACTORIES and Constructors for Cargo 174 Adding a Handling Event 175 Pause for Refactoring: An Alternative Design of the Cargo AGGREGATE 177 MODULES in the Shipping Model 179 Introducing a New Feature: Allocation Checking 181 Connecting the Two Systems 182 Enhancing the Model: Segmenting the Business 183 Performance Tuning 185 A Final Look 186 Part III Refactoring Toward Deeper Insight 187 Chapter 8: Breakthrough 193 Story of a Breakthrough 194 A Decent Model, and Yet . . . 194 The Breakthrough 196 A Deeper Model 198 A Sobering Decision 199 The Payoff 200 Opportunities 201 Focus on Basics 201 Epilogue: A Cascade of New Insights 202 Chapter 9: Making Implicit Concepts Explicit 205 Digging Out Concepts 206 Listen to Language 206 Scrutinize Awkwardness 210 Contemplate Contradictions 216 Read the Book 217 Try, Try Again 219 How to Model Less Obvious Kinds of Concepts 219 Explicit Constraints 220 Processes as Domain Objects 222 SPECIFICATION 224 Applying and Implementing SPECIFICATION 227 Chapter 10: Supple Design 243 INTENTION-REVEALING INTERFACES 246 SIDE-EFFECT-FREE FUNCTIONS 250 ASSERTIONS 255 CONCEPTUAL CONTOURS 260 STANDALONE CLASSES 265 CLOSURE OF OPERATIONS 268 Declarative Design 270 Domain-Specific Languages 272 A Declarative Style of Design 273 Extending SPECIFICATIONS in a Declarative Style 273 Angles of Attack 282 Carve Off Subdomains 283 Draw on Established Formalisms, When You Can 283 Chapter 11: Applying Analysis Patterns 293 Chapter 12: Relating Design Patterns to the Model 309 STRATEGY (A.K.A. POLICY) 311 COMPOSITE 315 Why Not FLYWEIGHT? 320 Chapter 13: Refactoring Toward Deeper Insight 321 Initiation 321 Exploration Teams 322 Prior Art 323 A Design for Developers 324 Timing 324 Crisis as Opportunity 325 Part IV Strategic Design 327 Chapter 14: Maintaining Model Integrity 331 BOUNDED CONTEXT 335 Recognizing Splinters Within a BOUNDED CONTEXT 339 CONTINUOUS INTEGRATION

341 CONTEXT MAP 344 Testing at the CONTEXT Boundaries 351 Organizing and Documenting CONTEXT MAPS 351 Relationships Between BOUNDED CONTEXTS 352 SHARED KERNEL 354 CUSTOMER/SUPPLIER DEVELOPMENT TEAMS 356 CONFORMIST 361 ANTICORRUPTION LAYER 364 Designing the Interface of the ANTICORRUPTION LAYER 366 Implementing the ANTICORRUPTION LAYER 366 A Cautionary Tale 370 SEPARATE WAYS 371 OPEN HOST SERVICE 374 PUBLISHED LANGUAGE 375 Unifying an Elephant 378 Choosing Your Model Context Strategy 381 Team Decision or Higher 382 Putting Ourselves in Context 382 Transforming Boundaries 382 Accepting That Which We Cannot Change: Delineating the External Systems 383 Relationships with the External Systems 384 The System Under Design 385 Catering to Special Needs with Distinct Models 386 Deployment 387 The Trade-off 388 When Your Project Is Already Under Way 388 Transformations 389 Merging CONTEXTS: SEPARATE WAYS ? SHARED KERNEL 389 Merging CONTEXTS: SHARED KERNEL ? CONTINUOUS INTEGRATION 391 Phasing Out a Legacy System 393 OPEN HOST SERVICE ? PUBLISHED LANGUAGE 394 Chapter 15: Distillation 397 CORE DOMAIN 400 Choosing the CORE 402 Who Does the Work? 403 An Escalation of Distillations 404 GENERIC SUBDOMAINS 406 Generic Doesn ’ t Mean Reusable 412 Project Risk Management 413 DOMAIN VISION STATEMENT 415 HIGHLIGHTED CORE 417 The Distillation Document 418 The Flagged CORE 419 The Distillation Document as Process Tool 420 COHESIVE MECHANISMS 422 GENERIC SUBDOMAIN Versus COHESIVE MECHANISM 424 When a MECHANISM Is Part of the CORE DOMAIN 425 Distilling to a Declarative Style 426 SEGREGATED CORE 428 The Costs of Creating a SEGREGATED CORE 429 Evolving Team Decision 430 ABSTRACT CORE 435 Deep Models Distill 436 Choosing Refactoring Targets 437 Chapter 16: Large-Scale Structure 439 EVOLVING ORDER 444 SYSTEM METAPHOR 447 The “ Naive Metaphor ” and Why We Don ’ t Need It 448 RESPONSIBILITY LAYERS 450 Choosing Appropriate Layers 460 KNOWLEDGE LEVEL 465 PLUGGABLE COMPONENT FRAMEWORK 475 How Restrictive Should a Structure Be? 480 Refactoring Toward a Fitting Structure 481 Minimalism 481 Communication and Self-Discipline 482 Restructuring Yields Supple Design 482 Distillation Lightens the Load 483 Chapter 17: Bringing the Strategy Together 485 Combining Large-Scale Structures and BOUNDED CONTEXTS 485 Combining Large-Scale Structures and Distillation 488 Assessment First 490 Who Sets the Strategy? 490 Emergent Structure from Application Development 491 A Customer-Focused Architecture Team 492 Six Essentials for Strategic Design Decision Making 492 The Same Goes for the Technical Frameworks 495 Beware the Master Plan 496 Conclusion 499 Appendix: The Use of Patterns in This Book 507 Glossary 511 References 515 Photo Credits 517 Index 519

章节摘录

插图：（The service asks each Net for assigned Rules（）, and then writes them fully expanded.）Of course, if there were only one operation （as in the example）, the script-based approach might be just as practical. But in reality, there were 20 or more. The MODEL-DRIVEN DESIGN scales easily and can include constraints on combining rules and other enhancements.The second design also accommodates testing. Its components have well-defined interfaces that can be unit-tested. The only way to test the script is to do an end-to-end file-in/file-out comparison.Keep in mind that such a design does not emerge in a single step. It would take several iterations of refactoring and knowledge crunching to distill the important concepts of the domain into a simple, incisive model.Letting the Bones Show: Why Models Matter to UsersIn theory, perhaps, you could present a user with any view of a system, regardless of what lies beneath. But in practice, a mismatch causes confusion at best——bugs at worst. Consider a very simple example of how users are misled by superimposed models of book- marks for Web sites in current releases of Microsoft Internet Explorer.

媒体关注与评论

“这本书应该出现在每个软件开发人员的书架上。
”　　——KerltBeck 软件开发方法学泰斗。
极限编程的创始人“Eric的这本书太棒太神奇了，他准确地告诉你如何让软件设计满足你的模型需求本书读起来趣味无穷。
Eric有许多有趣的故事，而且描述起来很有一套它出版后将成为软件开发。
人员必读的经典之作。
”　　——Ralh Johnson《设计模式》的作者“如果你认为自己在面向对象编程中的投资没有收到回报，那么读了本书你就会知道自己漏掉了什么。
”　　——Ward Cunningham 设计模式和敏捷软件方法的先驱“Enc Evarls力证作为开发核心的领域模型的重要性。
他搭建了一个稳固的框架并提供了一套实现技术和技巧。
这里沉淀下来的是亘古不变的智慧精华，在应时的方法论都沦为明日黄花后，它依然光华璀璨。
”　　——Dave Collirls Designing Object-Oriented User Interfaces的作者“Eric完全从实战者的角度着笔。
描述了无处不用的语言、与用户共享模型的好处、对象生命周期的管理、深度重构的过程和结果。
这是对我们这个领域的巨大贡献。
”　　——Luke Hollmann Beyond Software Architecture的作者

<<领域驱动设计>>

版权说明